COMP 504: Graduate Object-Oriented Programming and Design

Object-oriented programming is the major software paradigm used in large systems today. This includes most desktop applications such a office productivity software, data management applications and even the main frameworks of many games. But object-oriented programming and design (OOP/D) absolutely rules in the enterprise-class arena, the globe-spanning, mission-critical systems that, via the Internet, tie together international corporations from one end of the Earth to the other. In Comp504, we will discover how to use state-of-the-art object-oriented programming and design techniques to create flexible and scalable software systems that can interact with each other from anywhere in the world. We will learn about software design patterns and how they are used in multiple programming paradigms. Through a combination of theory and coding, the class will explore how highly decoupled systems with dynamically configurable behaviors are designed and implemented. The class will learn how to use design patterns as both a means of expressing fundamental computer science concepts as well as the building blocks in an abstract decomposition of a complex problem. In addition, the class will learn to use industry standard tools and technologies such as the Eclipse integrated development environment, Subversion source control and round-trip engineering design tools.

The course format will be that of lectures covering the theoretical, engineering and technological aspects of object-oriented programming and design that will be implemented in the project-like homework assignments. These homework projects will build upon each other as much as possible, culminating in a networked application.

Prerequisites: Comp201, Comp211, Comp212 or Comp215 or equivalent. Students should already be familiar with the full Java syntax and have experience writing procedural programs and algorithms in Java including the basic usage of classes, objects and inheritance as well as simple generics.

Primary target audience: Masters and Ph.D. CS students who are interested in object-oriented programming and design, large flexible systems, enterprise-class networked applications, dynamically configurable systems and software engineering. Students who plan on taking Comp410 in the future are strongly advised to take Comp504 beforehand. Taking Comp505 after Comp504 but before Comp410 is also strongly recommended. Graduate students may only take Comp504 (if they did not take Comp310 as an undergraduate). Graduate students may not take Comp310 which is only for undergraduates.

For more and the latest information, please see the Info page or contact Dr. Wong (swong at rice.edu)

CLASS TIMES: MWF 10:00 AM - 10:50 AM

LAB TIMES: Tues. 4:00 PM- 5:15 PM

MAIN CLASS WEBSITE: http://www.clear.rice.edu/comp504

Tentative Topic List

(in no particular order, not all topics may be covered due to time constraints)

Theory

- The Modeling of Complex Systems
  - Modeling only in terms of behaviors and not data
  - Modeling relationships in terms of *is-a* (inheritance)and *has-a* (composition)
  - The difference between abstract and universal representations
    - The problem with hoisting
  - Encapsulation
    - Classes, methods, and abstraction as encapsulation methods
    - State vs Strategy pattern
  - The 4 pillars of software abstraction
    - Abstract structure (abstract classes)
    - Abstract behavior (abstract methods)
    - Abstract construction (factory classes or methods)
    - Abstract environment (closures)
- Design Patterns (see list below)
  - Design patterns as embodiments of fundamental CS principles.
  - Appropriate and inappropriate uses of design patterns.
  - How design patterns manifest themselves in other paradigms.
- Lambda Calculus -- functional programming concepts in OOP/D
  - Commands vs. lambdas vs strategies
    - Execution of behaviors with and without known semantics
  - Lambdas vs. inner classes -- the role of closures
- The goals of a software system: extensibility, flexibility, robustness and correctness
  - Scalability as part of extensibility and flexibility

Highly decoupled systems:

- Why we want decoupled systems.
- How abstraction decouples systems
- Delegation model programming
- Using various design patterns to decouple system
- Command-driven systems and their relationship to message-passing systems.
- Dynamically configurable behaviors
  - Dynamic class loading for run-time modifiable behaviors
- Generics as an abstraction tool

Design patterns:

- *Union* - polymorphism
- *Null Object* - concrete representation of nothingness
- *Strategy* - abstract behavior with semantics
- *Command* - abstract behavior without semantics
- *State* - dynamic reclassification behavior

- *Composite* - recursive data structures
- *Visitor* - decoupled algorithms and data
- *Extended Visitor* - visitor with dynamically changeable hosts and algorithms
- *Decorator* - layered behaviors
- *Template* - abstract processes
- *Factory* - abstract construction
- *Adapter/Proxy* - abstract coupling
- *Model-View-Controller* (MVC) - decoupled display and computation
- *Observer-Observable* - decouples the message originator from the message receivers

Technologies

- Eclipse integrated development environment
- Subversion source control
- Unified Modeling Language (UML)
    - Class diagrams for static relationships
    - Sequence diagrams for dynamic relationships
    - Round-trip engineering between UML and code representations
- Remote dynamic class loading - automatic code transmittal across the web
- Remote Method Invocation (RMI) - executing behaviors on remote computers

Tentative Schedule

This schedule should only be considered as a guide. Expect that the schedule will change often! Also, the schedule will be affected by the timing of University recesses.

See the General Assignment requirements and instructions on how to use Subclipse/Subversion to turn-in your assignments.

Important Notice: While Comp504 shares many resources with Comp310, including lectures and assignment similarities, the requirements for Comp504 are NOT the same as for Comp310! The assignments and lectures for Comp504 are posted on this site only!

Lecture	**Labs (Tues)**	**Assignment Due Date (at beginning of class!) All COMM peer reviews due within 48 hrs.**
Lec01: Administrivia, Getting started	Lab01: Software installation, getting started with Eclipse and building GUIs with WindowBuilder	Complete your software installations BEFORE lab!
Lec02: GUI Building		
Lec03: Programming Paradigms		
Lec04: UML Diagrams Lec05: UML continued, Source Control & Ballworld Demos		
Lec06: HW01 review Lec07: Designing a Ball	Lab02: Animation	HW01: Simple GUI Program with Polymorphism COMM01: Programming Paradigms Comparison
Lec08: Connecting the Pieces		Comm01 Peer Reviews due in 48 hrs
Lec09: Composing Behaviors	Lab03: MVC Implementation	HW02: Inheritance-based Ballworld COMM02: Composites in the Real World

Lec10: Faking out the GUI with Factories		Comm02 Peer Reviews due in 48 hrs
Lec11: Factories, cont. & Transforming How We Paint	Lab04: Transforming Shapes	
Lec12: Using Superclasses as Service Providers		
Lec13: Painting continued and Command-Driven Dispatching		HW03: Composition-based Ballworld COMM03: Inheritance vs. Composition
		Comm03 Peer Reviews due in 48 hrs
Lec14: Command Dispatching	Lab05: Sending Commands	
Lec15: Collisions		
Lec16: Collisions, continued... Lec17: Ballworld Recap Lec18: Debugging with Eclipse and Visitors, revisited		HW04: Command-Driven FishWorld COMM04: Commands in the Real World
		Comm04 Peer Reviews due in 48 hrs
Lec19: Visitors... Presentation by Tubular Labs	Lab06: Collisions and Debugging	
Visitors, continued...		
Lec20: Extended Visitors		
Lec21: Extended Visitors Example - Self-balancing Trees	Lab07: Playing Music in Java	HW05: Command-dispatching Ballworld with Inter-ball Interactions COMM05: Modeling Abstract Behavior
Lec22: Extended Visitors Example, continued..., Music Project.		Comm05 Peer Reviews due in 48 hrs
Lec23: Insertion and Deletion Algorithms for Self-Balancing Trees.		

Lec24: Insertion and Deletion Algorithms for Self-Balancing Trees, continued... Lec25: Remote Method Invocation (RMI)		
Lec26: RMI continued...		HW06: ABC Music Player Comm06: Balancing Flexibility and Extensibility Against Security
		Comm06 Peer Reviews due in 48 hrs
Lec27: RMI Stubs Everywhere!	Lab08: Java RMI	
Lec28: Generic Extended Visitors and Data Packets		
Lec29: Use Cases and ChatApp API Design		
Lec30: Chat Program Interfaces	Lab09: ChatApp Design	HW07: Remote Task Execution Comm07: Scenarios for Extended Visitors
Lec31: ChatApp Interfaces... Subgroup API proposals completed and documented		Comm07 Peer Reviews due in 48 hrs
Lec32: ChatApp Design... Full API Proposal Documentation complete and Presentations		
Lec33: ChatApp finalization Revised Proposal Presentations and Final Voting	Lab10: ChatApp Design, continued & Simple Threads	
Lec34: ChatApp finalization, continued.. Revised Proposal Presentations and Final Voting, continued...		Testing session TBA

Lec35: Design Post-Mortem		
Lec36: Final Project	Lab11: Final Project Design	
Lec37: Game specifics discussion and Sequence Diagramming		Project Pair selection due
Lec38: Final Project Milestone 1 prep		HW08: RMI Chat Program Comm08: Architectures in Perspective Project Proposals due in Piazza
		Comm08 Peer Reviews due in 48 hrs
Lec39: RMI Garbage-Collecting and Final Project Issues	Lab12: Final Project Design and Implementation	
Lec40: Mixed Data Dictionareis, WWJ Mapping Library and the map package		
Lec41: Game Issues		Final Project: Milestone 1 Comm09: Peer-review of Comm08 only.
Lec42: Final Project Development Issues	Lab13: Final Project Communications Testing	
Lec43: Final Project Testing and Issues Discussion		
Lec44: Tips & Traps and Team Work	Lab14: Final Project testing	
Lec45: Lessons from Labs		
Lec46: We Made It!	Last Day of Classes!	

Demo Day! (Instructions)		Final Project must be fully operational. Operational testing will NOT be done after the demo time!
Final Project Code and Documentation Submission		Final Project: Milestone 2

TEXT BOOKS:

Though none of the existing text books truly meet our needs and requirements, we like the following book and highly recommend it for the course.

- Head First Design Patterns, First Edition, by Eric Freeman and Elizabeth Freeman, O'Reilly, 2004. ISBN 0-596-00712-4.

We also have a few other recommendations.

- The Java Programming Language, 4th edition, by Arnold, Gosling, and Holmes, Addison-Wesley,
  James Gosling is the creator of Java. This is the definitive Java reference. However, this book does not cover all the topics of the course and will be used only as a reference for Java syntax and semantics.

- Design Patterns, Elements Of Reusable Object-Oriented Software, by Gamma, Helm, Johnson, and Vlissides, Addison-Wesley, 1995. ISBN 0201633612
  This is the bible of the pattern community. It predates Java. The authors are affectionately referred to as the Gang of Four (GoF). All examples are written in C++ and/or Smalltalk. It can be hard to read if you are not familiar with these two languages, though the ideas are language independent. The "pattern language" used in this book also requires a lot of getting used to. We have learned so much from this book, and we are still reading it and learning from it. There is a version on CD-ROM which is cheaper (and perhaps, more versatile). We strongly recommend it, especially to any student considering further work in object-oriented programming.

- Thinking in Java, 4'th ed., by Bruce Eckel, Prentice Hall PTR.
  Good OO treatment, with a little more detail on syntax than Gosling's book. Students often find it useful for "nuts-and-bolts" information on programming. The full text of earlier editions are available as a free download from www.bruceeckel.com. Also at the web site is his on-line only book: Thinking in Patterns with Java, which is recommended reading.

Homework

All homework will be done in teams of AT LEAST TWO students. Students are NOT allowed to work on assignments alone. An important aspect of this course is learning how to communicate with and work with a variety of other people who have different opinions on how a problem should be approached. Learning how to work in a team is a critical skill for one's career. Except for the final project, the teams will be randomly assigned by the staff.

Team assignments for each assignment will be posted on the SharePoint site on the corresponding communications assignment page. The same team works on the current coding assignment, communications assignment and review of the previous communications assignment. Each team is assigned 2-4 peer review assignments of the previous communications assignment. Those peer review assignments are displayed on the previous communications assignment page.

Assignments are listed on the schedule by their due date. All assignments are due *before* class time on their due date.

Homework turn-in:

The turn-in mechanism will be the student's individual Comp504 Subversion repository under the main Comp504 repository on the Rice SVN servers, using the Subclipse version control plug-in in Eclipse. Please see

- Instructions on how to use Subclipse/Subversion to turn-in your assignments.
- General instructions on using the Subclipse plug-in for Eclipse

GRADING:

The course grade will be based on: ***TENTATIVE***

- Homework Assignments (55% total)
- Communications Assignments (15% total)
- Design Activities (10% total)
- Final Project (20% = 5% Milestone 1 + 15% Milestone 2)

Assignments' due dates and rules will be clearly stated at the time they are given.

No make-up work will be given unless there is a legitimate excuse such as proof of medical emergency. All permissions must be requested and approved in writing (hard copy or e-mail) at least 48 hours the due date!

We reserve the right to modify the above weightings at any time.

LATE POLICY:

- No assignments will accepted late without express written permission from an instructor **BEFORE** the due date of the assignment. (See above) Excused lateness will not be penalized unless otherwise decided by the instructor (student will be notified at time of request).
- **Extension requests will be considered ONLY for the following reasons and are NOT automatically accepted:**
    - Illness or unexpected travel occurring during or extending into the second half of the time period between the previous assignment due date and the current assignment due date.
    - Pre-planned travel that covers more than half of the time period between the previous assignment due date and the current assignment due date. Must request extension at least 48 hours BEFORE travel!
    - Unexpected technical problems that prevent the submission of the assignment such as a network or SVN outage.
        - Note that broken computers or faulty software installations are not acceptable justifications because all work can be completed and submitted from the OEDK lab computers.
- The following will NOT be considered as valid reasons for extensions:
    - Excessive work in other classes.
    - *Any* extension request made less than 48 hours before a deadline unless for an emergency or unplanned event as detailed above.
    - Travel to conferences where one is gone for less than half of the assignment time period. This includes the preparation for such trips.
    - Starting late and/or discovering that the assignment is more difficult than anticipated. START EARLY!
    - Not being able to meet with or having difficulty with an assignment partner.
- Any extension request must be accompanied by proof of having already made a concerted effort to complete the assignment, including SVN commits and meetings with the staff. Extensions will not be considered if no work has been done already on the assignment.
- ***If a late, unexcused submission is accepted at all, which is solely at the instructor's discretion and which is NOT the default policy:***
    - The assignment will be penalized *at least* 5% of the total possible score per day on top of any normal markdowns, rounded *up* to the nearest day.
    - If more one week after the due date, the assignment will not be accepted and will be given an automatic "0" score.

Technical and Development Resources:

- Comp310 Java Resources
- SVN Turn-in Instructions (See Subclipse plug-in instructions above for how to do this from inside of Eclipse, using the information at this link.)
- Using Eclipse
    - Eclipse and plugin-in installation
    - Getting started with Eclipse
    - Using the Subclipse Subversion plug-in
- Dr. Wong and Dr. Nguyen's research papers on OOP/D: http://www.bandgap.cs.rice.edu/personal/adrice_swong/public/WebPages/research/default.htm

Related Course Materials:

Principles of Object Oriented Programming courses:

- Connexions modules: http://cnx.org/content/col10213/latest/
- Comp201: http://www.clear.rice.edu/comp201
- Comp202: http://www.clear.rice.edu/comp202
- Comp211: https://wiki.rice.edu/confluence/display/cswiki/s11
- Comp215: http://www.clear.rice.edu/comp215

HONOR SYSTEM:

Rice University has a student-run academic honor system. Each student has personal responsibility for honesty in their own academic conduct. Each student has the responsibility to report any suspected problems to the Honor Council in a timely fashion. During any Rice exam, the student is expected to write the honor pledge on the outside of the exam:

On my honor, I have neither given nor received aid on this examination.

The Computer Science Department takes the Rice Honor System very seriously and handles all violations accordingly. General rules of conduct concerning collaboration will be spelled out before assignments or exams are distributed (see for instance, above). When in doubt, contact an instructor before proceeding.

For this class, the following additional restrictions MUST be observed:

You may not look at solutions to assignments from past semesters or from other students. Furthermore, attempting to take credit for someone else's work by turning it in as your own constitutes plagiarism, as defined by the Rice Honor Code.

You may not make your work in the class available to others in any form during or after the class. In particular, you may not place your code anywhere on the Internet that is viewable by others.

For example, posting one's code to a publicly viewable GitHub or other sort of repository or web page would be in violation of the above restrictions.

For more information concerning the Rice Honor System, see:

- The Honor Council home page
- Rice Honor System Handbook

ADA:

Any student with a documented disability needing academic adjustments or accommodations is requested to speak with us during the first two weeks of class. All discussions will remain confidential. Students with disabilities should also contact Disability Support Services in the Ley Student Center.